AR-008-918

DSTO-TR-0059

Maintaining Fully Replicated Distributed Databases in the Presence of Network or Node Failure and Repair

S.J. Miller

DEPARTMENT ◆ OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Maintaining Fully Replicated Distributed Databases in the Presence of Network or Node Failure and Repair

S.J. Miller

Information Technology Division
Electronics and Surveillance Research Laboratory

ABSTRACT

Technical Report

This report addresses a number of problems associated with the recovery of fully replicated distributed database systems in the enventuality of site failures and network partitions.

Accesion For	
NTIS CRA&I	[✓]
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

Approved for public release

DSTO-TR-0059

DEPARTMENT OF DEFENCE

DEFENCE SCIENCE AND TECHNOLOGY ORGANISATION

Published by

DSTO Electronics and Surveillance Research Laboratory
PO Box 1500
Salisbury South Australia 5108

Telephone: (08) 259 7053
Fax: (08) 259 5619
© Commonwealth of Australia 1994
AR-008-918
July 1994

CONTENTS

FIGURES

ABBREVIATIONS

A	Available
AT	Availability Table
A_Flag	Available Flag
C_Flag	Copy Request Flag
CO_Flag	CoOrdinator Flag
CR_Flag	Crash Flag
DM	Data Manager
FD	Failure Detection
FDA	Fully Distributed Algorithm
FDDI	Fiber Distributed Data Interface
FRDP	Failure and Recovery Detection Protocol
LAN	Local Area Network
N	Not available
NA	Node Availability
NI	Node Identifier
NM	Network Manager
PPN	Primary Partition Number
RD	Recovery Detection
RM	Reliability Manager
R_Flag	Recover Flag
SPN	Secondary Partition Number
TH	Transaction Handler
TM	Transaction Manager
TP	Transaction Processor
td	time delay
ti	time period

1 INTRODUCTION

The purpose of this report is to high-light some of the difficulties associated with the maintaining of a consistent real-time replicated database in the Naval combat system environment. For this reason, the protocol developed here, a modified version of Ma's Failure and Recovery Detection algorithm (FRDP), is used only as an example of the complex issues involved.

A number of distributed database architectures have previously been identified as having possible application in the single platform Naval combat system arena(1,2). These distributed database architectures have to meet the stringent requirements of the real-time environment, as well as be reliable, and easily maintainable in the eventuality of partial failure or battle damage. One important requirement is that the database system should remain available, even if in a somewhat degraded form, when subject to node failure or network damage. To meet these requirements, it is necessary for the database systems to have data replicated, either fully or partially, over the network nodes.

Distributed database systems, which have data replicated over a number of nodes (nodes often referred to as sites), must provide a mechanism for maintaining system functionality and data availability in the events of network partitioning, node failure, and subsequent repair. Such events must be detected early and appropriate crash, failure, or recovery algorithms performed. Recovery from such failures is complex and expensive in processing time, communications, and data storage. In order to maintain a globally consistent, replicated, and distributed database, complex recovery algorithms must be performed. These algorithms must have access to information relating to the network configuration, including database views prior to the failure, and transaction data logged since the failure. Recovery must be synchronised with transaction processing to ensure database consistency is maintained.

In some instances, since recovery from network partitions is difficult and has considerable impact on the system performance during the recovery process, it may be better not to implement an automatic partition recombining procedure. Instead, nodes on the secondary partition could be forced to recover as if they had failed independently, and only after any critical processing period (prosecution of threat evaluation and engagement functions), has passed, so that the possible loss of any data, or the effect on system performance, is minimised.

A number of database recovery methods have been described in the literature(3,4,10). These include the relatively simple node recovery process using a database Copy_Request and the more difficult partition recovery using a datapatch procedure, benchmarking with log merging, or rollback using local log transformation followed by the merging of the logs from each partition. It is not intended to compare here the different recovery mechanisms required for node and partition recovery as these issues have been addressed by a number of authors elsewhere(3,4,5,9,10,11). Instead, types of distributed network failure and subsequent recovery mechanisms are discussed, and a protocol for the detection of node failures, network partitions and their subsequent recovery, is described. This protocol is a somewhat simplified adaptation of the FRDP, developed by Ma(5). The modified protocol uses a coordinator for each partition and therefore does not have the complexity required for maintaining guardians. Further, by implementing a notification mechanism for node recovery, it distinguishes between network partition recovery and node failure recovery and treats them differently.

Section 2 describes the main failure conditions associated with a distributed network and Section 3 addresses node and network recovery. An FRDP is then described in the following three sections. In Section 4 an outline of the associated data structure needed to support the protocol is presented. Section 5 describes the protocol in detail. Finally, Section 6 addresses the application of the protocol to a fully replicated distributed database system(6) which uses database benchmarks and transaction time logs for network partition failure and recovery.

2 DISTRIBUTED NETWORK FAILURE

The availability, reliability, and performance of a distributed database system can be affected by the following failure situations:

a. Node failure,

b. Network partition, and

c. Message loss.

2.1 Node Failure

A node failure will occur if a node processor fails. The failure could be due to a hardware fault or a software error condition. In either case, no further local processing occurs, and there is no response to external messages.

2.2 Network Partition

There are two basic types of network partition. The simplest form of partition is the single node partition where a communication interface fails leaving the local processor isolated but still able to perform local processing. A more general type of partitioning occurs when a communication medium is severed (in the case of a Fiber Distributed Data Interface (FDDI) dual ring Local Area Network (LAN), broken in two places), leaving one or more nodes isolated from the rest of the network. In this situation, each portion of the network can continue to operate independently; however, the initially consistent database replications on each segment will gradually diverge.

2.3 Message Loss

Error checking and reliable processes for message delivery can be built into the LAN hardware and associated software to ensure messages reach their destinations correctly. When a broadcast transmission mode is used, there is normally no requirement for the receiving node to reply. In this case it is not possible to determine how many nodes did not receive the message due to a faulty interface. Further, an FDDI LAN, although less vulnerable to interference and hence having the potential to be a more reliable network, can loose frames during a network re-configuration(8). These data losses could be detected if nodes were required to reply to broadcast messages. Then if a reply is not received from one or more nodes, the message can be re-broadcast. However, it would be necessary to decide, in the event of continued missing replies, how many times a message should be re-broadcast before a failure condition is recognised. Also, a sequence number would need to accompany the messages so that nodes can identify re-broadcasts and reply to them without acting on them a second time.

Even when a message acknowledgment procedure is implemented, if a node failure or network partition occurs during a broadcast, and the originator does not receive a reply, it still can not be certain whether this was because the broadcast message was not received by a remote node, or because the reply was lost. In such a situation the actual state of the remote node's distributed database replication would be uncertain.

3 DISTRIBUTED NETWORK RECOVERY

A distributed processing system must be able to recover from the consequences of lost messages, network partitions, and node failures. During the recovery process, all the distributed database replications must be made consistent with each other and the replicated data brought into agreement. Databases at the recovering nodes are relatively easy to restore; however, making the databases on different network partitions consistent on repair, when two network partitions are recombined, is much more difficult. Further, the detection of database differences resulting from lost messages requires considerable additional database management overheads and results in increased LAN message traffic. A reliable database update process, using handshaking communication to ensure messages are reliably sent to all nodes, can reduce this overhead.

It is impossible for a local node to distinguish between a remote node failure and a network partition. The only way a node can detect the occurrence of either situation is by means of message timeouts. A suitable time delay must be determined if message timeouts are to be realistic in the real-time environment. This delay must take into consideration parameters such as the processing delays and communication delays. At recovery, however, the failure situation can be resolved. A recovering node can inform the network or its partition when it is again serviceable and ready to re-join the network. This provides the mechanism for a recovery algorithm to distinguish between a node failure and a network partition.

3.1 Node Recovery

The requirements of a node recovering from a failed condition, and those of a new node joining the network or locally available partition, are similar. In both cases, the node must be initialised and a new copy of the distributed database obtained. This is the simplest of all the recovery situations as there would have been no database activity and hence no possible impact on the global database prior to the joining of a new node or the re-joining of the recovering node. Both these cases can therefore be treated as a recovery situation. Bhargava(9), states that when data items are brought into the database separately by independent copiers (special transactions), the recovery algorithm must "guarantee that no user transaction can read a copy at the recovering node before the copy is renovated, and once a copy has been joined into the database, all transactions writing to the logical data item also contained in the data copy, must update this copy as well."

A recovering node must perform one or more "Copy_Requests" (requests for copies of data), to obtain a current copy of the database. Copy_Requests must be treated in a reliable manner similarly to that of an update. They must be serialised using a concurrency control protocol at the node supplying the data copy. No database record is to be read at the recovering node until it has been written to by a Copy_Request reply (ie the database partition containing the record to be written to, has been copied to the recovering node). During the copying phase, all updates which require access to records on the data partition at the recovering node, must be logged in timestamp order for execution after the data partition has been recovered. Further, no transaction is to write into a new node record until it has been fully restored. This could be ensured by using the guardian approach(3) (see Section 3.1.3).

The implementation of the Copy_Request depends upon the database architecture and the database granularity (eg. a page or record structure). If data is stored in pages and these are replicated in the form of an Owner and a number of Subscribers, then a Copy_Request can be broadcast and only the Owner will be expected to reply with a new page copy to the requester. However, if an Owner and Subscriber mechanism is not used, and a fully replicated database architecture exists, then the Copy_Request can be implemented in a number of ways, two of which are:

a. Broadcast

b. Nearest node

If either of the participating nodes fail during the recovery process, then the recovery process must be aborted and any updates logged at the recovering node deleted.

3.1.1 Broadcast method

Node recovery using the broadcast method is a heavy user of communications. It involves broadcasting a sequence of Copy_Requests to all nodes in the global network and processing the first sequence of replies received. Each node receiving the requests, provided it is not itself recovering, reads its local database copy, formats a sequence of replies containing the portions of the database requested, and posts it to the requesting node. The first sequence of replies reaching the requester is accepted and used to update its local database. All subsequent reply sequences, even though occupying communication bandwidth, are ignored.

3.1.2 Nearest Node method

A better approach to node recovery is for the node to keep track of the available nodes in a network partition using a simple node availability algorithm, and request a copy of the data from the nearest available node. A node is considered available if it is active and not currently in the process of recovery. The recovery process requires the recovering node to first perform an availability algorithm to obtain local knowledge of all the available nodes, and then post a sequence of Copy_Requests to the nearest node, either the next lowest or next highest numbered available node as determined by a simple algorithm. Very little overhead would be required to keep an updated list of the available nodes. The data accompanying the Copy_Request replies is used to update the recovering node's database.

3.1.3 Guardian approach

The guardian approach(3), can be used to ensure a logical data item at a new node is not updated until it has been written to by the recovery process. For each logical data item at the new node, designate a "guardian copy" at an old node. After the new node is added, the node holding the guardian copy alerts all transactions that updates should write into the new copy also.

3.2 Recovering From a Network Partition

Recovering from network partitions requires the participation of both the primary partition and the re-joining partition. The transactions which have occurred on each of the separate network partitions since the network partitioning, must be applied to the other re-joining partition in a manner which makes all the separate database replications on each partition consistent with each other, without the loss of any data.

There are a number of methods available for recovering from network partitions and making data in replicated databases consistent. These include the log transformation method(4), and the precedence graph method(11). A simpler method, but one which requires different views of the database at each node, is worth considering. This method uses bench marking, by saving the current state of the database when a partition is recognised, and subsequently keeps a log of all local transactions applied to the node's active database. The log can be merged with a similar log from the network partition and applied to the local bench mark database upon recovery. This method has the drawbacks of requiring each available node to keep a second view of the database, and a relatively large log storage when partitions exist for long periods of time.

With all types of recovery mechanisms, the first step is to recognise the presence of and the recovery of a partition. There have been a number of FRDP's proposed in the literature. Ma's FRDP, consisting of two protocols, the Failure Detection (FD) and the Recovery Detection (RD) protocols(5), is typical. In the following sections a variant on Ma's FRDP, which allows node recoveries to be distinguished from network partition recoveries, is presented.

4 DATA STRUCTURE REQUIRED FOR THE FRDP

Before presenting the proposed FRDP, the data structures used by the protocol must be defined. Each node must maintain a table showing the local network view and a set of flags whose states determine the node's operational state. Each node will need to be kept informed of the dynamic state of the network by means of both data and special purpose control messages. All the messages will include timestamps to provide correct event ordering. The timestamps will also be used to maintain local clock synchronisation.

4.1 Availability Table

Each node on the network will be provided with a unique Node Identifier (NI). The NI for nodes will be an integer value between 1 and m where, m represents the maximum number of nodes to be attached to the network. A node Availability Table (AT) will be maintained by each node. This table will keep track of the current local view of the network. It will contain up to a maximum of m sets of node entries. Each set will consist of an NI, Node Availability (NA) ("A" for Available, "N" Not Available), a Primary Partition Number (PPN), and a Secondary Partition Number (SPN) as shown in figure 1 below:

NI	NA	PPN	SPN
01	A (available)	0	0
03	A	0	0
04	N	0	1
I	0	I	0
m	0	I	0

Figure 1. Node Availability Table

The PPN indicates to which primary network partition each available node belongs while, the SPN lists the secondary partition/s to which the non-available nodes belong. The allocation and the maintenance of the partition numbers is discussed in the Appendix.

4.2 Control Flags

Each node will also maintain the following five flag variables.

a. An Available flag (A_Flag); to indicate the node is available for transaction processing,

b. a Recover flag (R_Flag); to indicate when the node is part of a network recovery,

c. a Copy request flag (C_Flag); to indicate when a copy of the database is being transferred to another node,

d. a Crash flag (CR_Flag); indicating a crash algorithm is currently running, and

e. a CoOrdinator flag (CO_Flag); indicating whether the node is currently a coordinator.

4.3 Logical Clock

Timestamp based concurrency control requires transactions to be serialised in time order. This time ordering must be globally controlled across all nodes in the distributed system and can be achieved by means of timestamps (time_now) generated from a global clock which may comprise of real clocks or logical clocks, one at each node. These node clocks must be synchronised closely so that events can be totally ordered across the network. The maintaining of clocks and ordering of events is a subject on its own and has been well developed by Lamport(12).

It is proposed here that each node should have a local logical clock and all messages should include a timestamp (time_now) derived from this clock. Each time a message is received at a node a synchronisation algorithm should be initiated. The local clock is to be set equal to the greater of the local time_now and the external node time_now value accompanying the message. This ensures all node clocks are kept in close synchrony with the fastest node clock on the network.

4.4 Messages

A number of control/data messages are also required to maintain a local view of the network and control the recovery process. These messages include:

- IM UP
- COPY REQUEST
- NODE
- RECOVER

- REORG

The above messages, except for NODE and REORG, expect an acknowledgment in reply from the appropriate coordinator. All messages will include a header portion containing time_now, the source's NI, and a PPN value. The NODE header will include the source's PPN while the RECOVER header will include the PPN obtained from the NODE.

The IM UP acknowledgment, the RECOVER and the REORG will also include a copy of the source's (a coordinator) AT as data.

4.4.1 IM UP

An IM UP message will be broadcast by a recovering or new node wishing to join the network. Only a coordinator can acknowledge receipt of this message. It does this by replying with a copy of its current AT. On receipt of the acknowledgment, the new node sets both its A_Flag and C_Flag, and issues a number of Copy_Request messages (COPY REQUEST) to obtain a copy of the database.

All other nodes receiving the IM UP, update their ATs to reflect the availability of the recovered node. This is done by including an entry for the new NI, setting the associated NA to "A", and setting its associated partition numbers PPN and SPN to reflect the coordinator's view of the local partition status.

4.4.2 COPY REQUEST

A new node, in response to an IM UP reply, will initiate a transaction sequence of Copy_Requests. Each Copy_Request reads the appropriate portion of the database and formats and posts a COPY REQUEST message to the identified coordinator. A sequence parameter indicating whether the request is the "first", an "intermediate", or the "last" COPY REQUEST in the sequence, and a parameter identifying the portion of the database involved, will be included as part of the message. A coordinator, in response to the "first" COPY REQUEST, must set its C_Flag and reply with a copy of the relevant database portion. During the copying process, all new transactions at both the new node and the coordinator must be logged for execution after the process has completed. When the database copying process is complete and the last data portion is copied, the C_Flags at both the new node and the coordinator node must be cleared to allow normal processing to continue. This tight control is necessary to provide a reliable database transfer from the coordinator to the new node. Without proper control a transaction could update a partially transferred database without propagating the change to the new database copy. This would result in global database inconsistencies.

4.4.3 NODE

NODE messages will be broadcast periodically by each node. They will be used to identify when a node/partition is available to rejoin a primary network, and to provide a means of synchronising node checks for coordinator eligibility status. Each time a node receives a local NODE message it must compare its NI value with the NI values of all the other available nodes listed in the AT. If the receiving node's NI is smaller than all the other available node' NIs listed in the AT, it will assume the coordinating role and set its CO_Flag true.

4.4.4 RECOVER

When a NODE message is received from a node on the recovering secondary partition, the primary network coordinator, provided it is not currently involved in a recovery, will issue a RECOVER.

Any node which is not currently performing a recovery or crash process, having neither their R_Flag or CR_Flag set, can receive and process the RECOVER. However, only another coordinator on the recovering secondary partition can acknowledge it and actually initiate the recovery of that partition (see Section 6.2.4). During the recovery phase all new transactions must be logged for execution after the recovery is complete.

Each node which receives the RECOVER must update its local AT (those items associated with nodes on the primary network segment only), and set its R_Flag ready to participate in the recovery process.

If the node is on the primary network (eg a node with a PPN matching the left-hand portions of the PPN accompanying the RECOVER), it must identify the database benchmark from the right-hand portion of the PPN accompanying the RECOVER.

If the node is on the secondary network (eg a node with a PPN matching the right-hand portion of the PPN accompanying the RECOVER), it must identify the database benchmark from the left-hand portions of the PPN accompanying the RECOVER.

Further, if the node is a secondary coordinator, it must identify the appropriate log, and send an acknowledgment back to the primary source. The acknowledgment could be a data message containing a copy of the coordinator's relevant log data.

4.4.5 REORG

A secondary coordinator will issue a REORG during a recovery process after all log data has been exchanged, to update all local ATs and initiate the recovery algorithm at each node (Section 5.4.2).

Any node which receives a REORG message and has its R_FLAG set, currently participating in a recovery process, must update its local AT, and initiate a recovery algorithm. After the recovery process is complete the R_Flag must be cleared. All new transactions must be logged during the recovery process for later processing.

5 THE FRDP PROTOCOL

The FRDP has been designed to handle both the detection of node failures and network partitions, as well as the detection of their subsequent recovery. Each of these situations can occur singly or in any multiple combination. The limitation being the ability of crash and database recovery algorithms to handle the combinations due to the restrictions set by available processing time, communication speed and available data storage.

5.1 Normal protocol operation

Each node has a unique NI allocated for identification purposes. The overall view of the current network or network partition is maintained in an AT at each node (Section 4.1). A normal operating mode exists when the A_Flag is set and the R_Flag, the CR_Flag, and the C_Flag are all cleared. In this state, a node can receive data, transactions, control messages, identification messages, and frequently check its eligibility for the coordination role. During normal operation, each node must periodically broadcast a NODE message.

Each time a NODE message is received, a node (coordinating node included) must check whether it has the lowest NI of all the available nodes in the partition (Section 4.4.3). A change of coordinator can only take place provided an initialisation process, a recovery, or a crash process is not in progress, ie none of the C_Flag, R_Flag, or CR_Flag are set. A coordinator node which no longer has the lowest NI must relinquish its coordinating role and clear its CO_Flag while, the node with the lowest NI, not currently a coordinator, will set its CO_Flag and take on the coordinating role.

A coordinator has the lead role in any network re-configuration resulting from a repair or failure condition, and is the only node which can send RECOVER messages or receive COPY REQUESTs.

5.2 Node joining the network

At startup (ie when a node is joining or re-joining the network), a node's local AT will be empty. The processing required to integrate the new node into the network will be determined by the current network configuration and the response obtained, if any, to an "IM UP" broadcast message. Typically the processing is as follows:

The new node broadcasts an IM UP request message and sets a timeout period, say t0 (see section 3), a time period during which the reply from the network coordinator can be accepted. This request is a notification to an existing network coordinator of the node's intention to join/re-join the network. An existing coordinator should respond with a copy of its global view of the network, ie a copy of its AT in acknowledgment. If the IM UP request times out (ie no coordinator response within the timeout period t0), then it should be resent after a suitable delay td. The delay introduced by td must be sufficiently large to allow any current coordinator election process to complete. A network re-organisation due to a loss of a coordinator may be currently in progress. If there is still no response, the new node must assume it is the only node on the network, and initialise its local AT with the inclusion of an NI listing for itself having NA, PPN, and SPN set to "A", "0", and "0" respectively. Finally, the new node should set both its A_Flag, and CO_Flag to initiate normal processing as a coordinator.

When a timely acknowledgment of the IM UP is received, the new node temporarily stores the accompanying copy of the coordinator's AT, sets both its A_Flag and C_Flag, and initiates a sequence of COPY REQUEST messages to obtain a copy of the coordinator's database (see Section 4.4.2). If the coordinator responds to the COPY REQUESTs within a timeout period eg t1, then the new node's database is initialised to match that of the coordinators'. If a partitioned network exists, the new node will also receive copies of all the coordinator's benchmarks and logs (see Section 5.2.1). After the database initialisation is complete, the local AT is initialised to reflect the network view. The view is developed from the temporarily stored copy of the coordinator's AT, with the inclusion of the new local node's NI, and the associated NA, PPN, and SPN values set to reflect the coordinator's view. Finally, normal node processing is initiated by clearing its C_Flag.

If there is no response to the "first" COPY REQUEST in the sequence (ie within the timeout period t1), then the new node assumes there is now no coordinator, it discards the temporary copy of the AT, takes on the coordinator role itself by initialising its local AT and setting both its A_FLAG and CO_Flag, as if there had been no response to the original IM UP request.

If there is no response to any subsequent COPY REQUESTS within a timeout period t1, the node must assume it has become isolated and restart the sequence with a new IM UP.

Once normal processing has commenced, a background activity on the new node will periodically broadcast a NODE message.

5.2.1 A coordinator's response to IM UP and COPY REQUEST messages

A coordinator has the responsibility of providing a new or recovering node with a view of the network configuration and a copy of the global database.

When a coordinator receives an IM UP it first checks its local AT for a matching NI with NA marked "N". If one exists, it clears any relevant no longer needed benchmark and log initiated when the currently recovering node failed, as the following COPY REQUEST process provides the new node with a consistent copy of the coordinator's database. The coordinator completes the IM UP message processing by posting an acknowledgment, containing a copy of its updated AT, to the recovering node (Section 4.4.1). The benchmark and log can be identified by the SPN associated with the matching NI.

When the coordinator receives a sequence of COPY REQUESTs, it sets its local C_Flag, and responds with a copy of each relevant database portion requested. After replying to the "final" COPY REQUEST, if the coordinator is in a partitioned network, it sends all the benchmark and log data to the new node. It finally clears its C_Flag and updates its AT to include the new NI with the associated NA, PPN, and SPN values set to reflect the coordinators new view of the network (Section 4.4.2).

If the "final" COPY REQUEST is not received within a timeout period t2 (t2 is the time from the "first" COPY REQUEST in the sequence), the process is aborted without the coordinator updating its AT.

5.2.2 A non-coordinator's response to an IM UP message

All nodes can receive IM UP messages. A non-coordinator node updates its AT to include the new node (Section 4.4.1). When a node receives an IM UP it checks its local AT for a matching NI with NA marked "N". If one exists, as this is not a recovering partition, it clears any relevant benchmark and log initiated when the currently recovering node failed. The benchmark and log can be identified by the SPN associated with the matching NI.

5.3 Node failure and network partition

Node failures and network partitions will be detected during reliable transaction operations, (see Section 6.3). A node which detects either of these situations must calculate, as appropriate, new PPN and SPN values (see Appendix), update its local AT, and finally, notify all the other available nodes of the network change. A crash algorithm must be performed on all the available nodes before the reliable update is completed (see Section 6.2).

5.4 Network Recovery

A network recovery procedure must take place when two partitions are to be re-united. This involves detecting the recovery and initiating a database recovery algorithm to merge together any data collected during the failure to form one consistent global database.

5.4.1 Recovery detection

A network recovery between two network partitions is detected when the primary partition coordinator receives a NODE message from a node on the secondary partition.

5.4.2 Recovery

The actual recovery process depends on the crash and recovery algorithms implemented. As an example, database benchmarking and crash logs with the subsequent merging during recovery of the logs and benchmarks from the rejoining partitions, has been used in an example application of the FRDP (see Section 6).

The coordinator on each partition controls the recovery process. When a primary network coordinator detects the recovery of a secondary partition (receipt of a NODE message from a secondary partition node), and it is not already involved in a recovery (R_Flag not set), it broadcasts a RECOVER message containing the PPN received with the NODE message and a copy of the coordinator's local AT. It also identifies the relevant benchmark and log from the right-hand portion of the PPN for later processing (Section 4.4.4). On receipt of an acknowledgment from the secondary partition coordinator, the primary coordinator sets its R_Flag, formats and sends a data message containing the relevant log data to the secondary coordinator.

When a coordinator on the secondary partition receives the RECOVER it updates the primary node NI data in its local AT (from the enclosed copy of the primary coordinator's AT), identifies the relevant log data from the enclosed PPN, and formats and sends a data message containing the log data in acknowledgment. It then waits for the log data message from the primary coordinator. On receiving the log data message from the primary coordinator, the secondary coordinator initiates a recovery algorithm locally and broadcasts a REORG message containing a copy of its updated AT to all other nodes.

All nodes can receive RECOVER and REORG messages. A node receiving a RECOVER checks both its R_Flag and C_Flag, if either is set no action is taken as a recovery process is already in progress. Otherwise, it performs the processing outlined in section 4.4.4 and waits for a REORG. When a REORG is received, and the R_Flag is set, the recovery procedure is initiated (Section 4.4.5). If a REORG is not received within a timeout period, say t3, then it must be assumed that the secondary coordinator has failed and the R_Flag must be cleared to abort the recovery process.

The recovery process must be reliable and sufficiently robust to handle both node and network partitioning while a recovery is actually in process. A timeout is employed in the early stages of a recovery to allow the process to be aborted if a failure involving a coordinator node occurs before the REORG message is sent. The recovery process is effectively handed over to each node to independently complete on receipt of the REORG. If a network or node failure occurs which does not include a coordinator, the recovery process will continue to completion normally without any knowledge of the new failure. The failure will be recognised and treated in the normal fashion only after transaction processing is again initiated.

6 APPLICATION OF THE FRDP

The application of interest here is the naval single platform combat system, a real-time application involving high rate input sensor data, once only operator entered data, and critical actions requiring attention within a set deadline. Such a system must be capable of functioning reliably in an hostile environment where action damage or other failures could occur. It must continue to operate under these conditions even if in a somewhat degraded form. One way of meeting this requirement is to implement a distributed processing system with a dynamically reconfigurable, fully replicated, and distributed database. It is in this type of system that a fully distributed update algorithm(6,7) combined with a variant of Ma's FRDP can be employed.

Neither Ma's FRDP or the variant proposed in Section 5 keep a current physical view of the network. Instead, they keep a logical view which is periodically updated. A new view of the local network partition can be deduced from the replies received during a Reliable update. If a recent network partition or node failure has occurred (ie since the last logical view was updated), then the new network view must be stored (local AT updated, see section 6.3.1), and a crash process implemented (Section 6.2). This sequence is essential if a globally consistent database is to be achieved by a recovery process at network repair.

The Transaction Manager (TM) and Data Manager (DM) model of the Fully Replicated Distributed Database System(6) employing Singhal's Fully Distributed Algorithm (FDA)(7) is assumed here. Implementation of the FRDP requires the addition of a Network Manager (NM) at each node. It is assumed that the managers cannot fail independently and that failures are clean. This means that when a node or component of a communication sub-system fails it simply stops running. These situations are considered fail safe, a condition where no malicious messages are sent or incorrect operations performed. A further assumption is that when a failed component recovers it knows that it has failed. A process failure occurs when the entire node fails.

The state of a node's TM determines the mode of operation of the database system at that node and at any time can be one of the following:

a. Normal; indicating normal transaction processing and may include updating of logs for unavailable nodes,

b. Failure; indicating a period during which a benchmark database is being taken and a log is being initiated by a crash algorithm, or

c. Recovery; indicating that the node is executing a merge algorithm.

During the failure and recovery modes, the application transaction processing must be halted and transactions queued for later processing to prevent changes to the database while database snapshots or a recovery is in progress.

A possible node functional architecture is depicted in figure 2. Here a modular approach has been taken where each node is required to implement the following common set of functions to maintain the availability and consistency of the fully replicated distributed database for the node applications:

- Communication,
- Network Management,
- Transaction Management, and
- Database Management.

6.1 Communication

The Communication's function is to provide an interface to the network. It processes control and data messages, which may be broadcast or sent point-to-point, for the TM and NM. Network control messages are received from, or sent to the NM while transaction requests and control messages, including the COPY REQUEST replies containing the external coordinator's database information, are channelled to the TM.

This task has also the responsibility of implementing the clock synchronisation. Each time a message is sent a timestamp derived from the local clock is attached. Timestamps from received messages are read and input into a synchronisation algorithm (Section 4.3).

6.2 Network Management

The NM has the prime role of performing network management and maintaining the local logical view of the network. It does this by sending control messages over the network, processing any received messages, and maintaining the local AT and control flags appropriately. The locally maintained logical view identifies the reachable and operational nodes. However, since the only way of detecting a failed or unreachable node is by transaction timeouts, some nodes listed as available could actually have failed since the last transaction. This is not a serious problem and will be corrected by a subsequent reliable transaction process, (see Section 6.3). When the NM is notified by the TM of a crash condition (crash notification message), it calculates a new SPN from the contents of the local AT for the failed sites, updates the AT, and initiates a local crash algorithm.

The crash algorithm takes a benchmark of the current database for the new SPN partition and sets up a log onto which the current transaction and future transactions can be stored while there are unavailable nodes, or until a recovery is initiated. When the crash process is completed the CR_Flag is cleared to allow the TM to continue normal transaction processing.

The NM is responsible for the issuing of, and processing of IM UP, NODE, RECOVER, and REORG messages (Section 4.4). It also has a significant role to play during both node initialisation and recovery (new node), and network repair. Periodically all node NMs will send a NODE message as a means of identifying its presence to the network, and check whether they have the lowest available NI (see Section 5.1).

6.3 Transaction Management

The Transaction Manager (TM), is responsible for the management of both local and external transaction processing. It must ensure that all transaction's reliability requirements, as in the case of the Reliable update, and a Copy_Request, are met. A Reliable update process is used for the once only entered data while, a Performance update is used for high input rate sensor type data(6,10). The TM, by using a reliability mechanism such as a two phase commit(13), is in the best position to recognise a recent network partition or node failure and can best control specialised crash and recovery processing. A functional description of a typical TM is presented in a previous paper(6). The TM consisted of a Transaction Handler (TH), a Reliability Manager (RM), a Transaction Processor (TP), and a duplicated Communication Interface.

The TM manages transactions during normal operation (eg when the A_Flag is set and the R_Flag, C_Flag, and CR_Flag are all cleared), and provides serialisation and reliability when appropriate. If any of the flags R_Flag, C_Flag, or CR_Flag are set, then application generated transactions from either the local or an external node, are queued for later processing. During normal operation the RM accepts the next (oldest) transaction from the TH for processing. The transaction could be either a local transaction or an external transaction requiring either performance or reliable processing. A Performance transaction is passed directly to the TP for completion while a Reliable transaction remains under the control of the RM and a variant of the two phase commit protocol employing Commit and Commit_Reorg messages. Since the emphasis is on database availability, the protocol has been modified to include the Commit_Reorg messages instead of the usual Abort messages(13).

6.3.1 Local Reliable Transactions

Local Reliable transactions can be managed by the RM using a two phase commit procedure in the following manner; The RM passes a copy of the transaction to the Communication Interface for broadcasting and then sets a timeout period during which acknowledgments from each of the listed available nodes are expected. On receipt of the correct number of acknowledgments, the RM formats a Commit message, sends it to the Communication Interface to be broadcast, and finally, passes the transaction to the TP for completion. If a timeout occurs before each node acknowledges the transaction, then the RM requests a rebroadcast of the original transaction. If a timeout occurs a second time, the RM sets the CR_Flag and performs the following processing in the order listed:

a. Updates the local AT to reflect the current logical view of the local network partition,

b. passes the transaction onto the TP,

c. formats a Commit_Reorg message for broadcasting to the available nodes, and

d. finally notifies the NM of the crash.

6.3.2 External Reliable Transactions

Only external Reliable transactions from the recognised local network partition are processed. The presence of an unexpected Reliable transaction (ie one from another partition), means that a network recovery is imminent (ie to be implemented on receipt of a NODE message). Unexpected Reliable transactions must be ignored as they would have been logged as part of the other partition's crash processing (section 6.2).

All external Reliable transactions emanating from the local partition are treated slightly differently by the RM than the local generated Reliable transactions. For these, the RM issues an acknowledgment, sets a time out period, and waits for a Commit or Commit_Reorg message.

When a Commit is received, the reliability aspects of the transaction are satisfied, and the transaction is passed onto the TP for completion.

If a Commit_Reorg is received, the RM must set the CR_Flag to inhibit further transaction processing while a crash process is implemented. The crash process involves the RM in updating the local AT from the AT data accompanying the Commit_Reorg to form a new local logical view of the network and then notifying the NM of the crash.

If a timeout occurs and neither a Commit or Commit_Reorg has been received it is assumed that a network failure or partition has occurred. In either case the RM sets the CR_Flag, passes the transaction onto the TP, and notifies the NM of the crash.

After a crash process is completed the CR_Flag is reset to allow the TP to resume normal transaction processing.

6.3.3 Copy_Requests

In addition to the normal application update transaction processing, the TM can at the local NM's request, issue a Copy_Request transaction and process the associated COPY_REQUEST replies. The TM on a coordinator node processes the COPY REQUEST transactions. It obtains a copy of the local database and passes it to the Communications interface for sending to the originator of the COPY REQUEST. The TM on the node originating the COPY REQUEST processes the COPY REQUEST reply in a similar manner as it would an application requested transaction. On completion of the database update it clears the local C_Flag.

6.4 Database Management

The DM performs the database management function. It ensures all reads and writes are performed serially in timestamp order and provides an interface to the local copy of the distributed database.

7 CONCLUSION

This paper has addressed the distributed database system issues of node and network failure and recovery. A protocol based on the FRDP proposed by Ma and Belford is described and an outline of how it could be applied to the Fully Replicated Distributed Database System Employing an FDA is presented. The application of the proposed FRDP supports the example of a crash algorithm implementing benchmarking and transaction logging and a merge algorithm for network partition recovery.

The proposed FRDP does not implement any guardians. Instead, it maintains dynamically one partition coordinator for each resulting network partition. As a result of this simplified approach, it has been possible to include a mechanism for identifying a failed node's recovery, and to separate the more complex, processing intensive network partition and recovery functions, from the relatively straight forward node recovery situation. A node recovery is treated the same as a new node.

The protocol presented here is more flexible and therefore more suited to the Naval combat system environment than is Ma's protocol. Further, it would be possible to only implement the failure and new node/node recovery portions of the protocol if system performance was a major issue and a reliable and fault tolerant communication system such as FDDI-II is available.

REFERENCES

No.	Author	Title
1	Schapel, J.G.	"A Survey of Distributed Processing Architectures Suitable for Combat Systems". WSRL-TM-8/90, September 1990
2	Miller, S.J.	"Trends in Distributed Processing for Naval Combat Data Systems". WSRL-TM-22/89, February 1989
3	Attar, R. Bernstein, P.A.	"Site Initialisation, Recovery, and Backup in a Distributed Database System". Proceedings of the Sixth Berkeley Workshop on Distributed Data Management and Computer Networks, Feb 1982
4	Blaustein, B.T. Garcia_Molina, H.	"Maintaining Replicated Databases Even in the Presence of Network Partitions". IEEE Eascon, 16th Conference, Sep 1983
5	Ma, A.V. Belford, G.G.	"A Failure and Recovery Detection Protocol for Optimistic Partitioned Operation in Distributed Database Systems". IEEE Computer Society, 6th International Conference on Distributed Computing Systems, May 1986
6	Miller, S.J.	"A Fully Replicated Distributed Database". ERL-0719-RN, June 1993
7	Singhal, M.	"Update Transport: A New Technique For Update Synchronisation in Replicated Database Systems". IEEE Transactions on Software Engineering, Vol. 16, No. 12, Dec 1990
8	Ross, F.E.	"FDDI - A Tutorial". IEEE, Communications Magazine, May 1986, Vol. 24, No. 5
9	Bhargava, B. Ruan, Z.	"Site Recovery In Replicated Distributed Database Systems". IEEE, Distributed Computing Systems 6th International Conference 1986
10	Reilly, M. Tillman, P.R.	"Maintaining Consistency and Accommodating Network Repair in a Self-Regenerative Distributed Database Management System". AGARD-CP-360, Feb 1985

11	Davidson, S.	"Optimism and Consistency In Partitioned Distributed Database Systems". ACM Transactions on Database Systems, Vol. 9, No. 3, September 1984
12	Lamport, L.	"Time Clocks and Ordering of Events in Distributed Systems". Communications, ACM Jul 1978
13	Ceri, S. Pelagetti, G.	"Distributed Database Principles and Systems". McGraw-Hill book company, 1984.

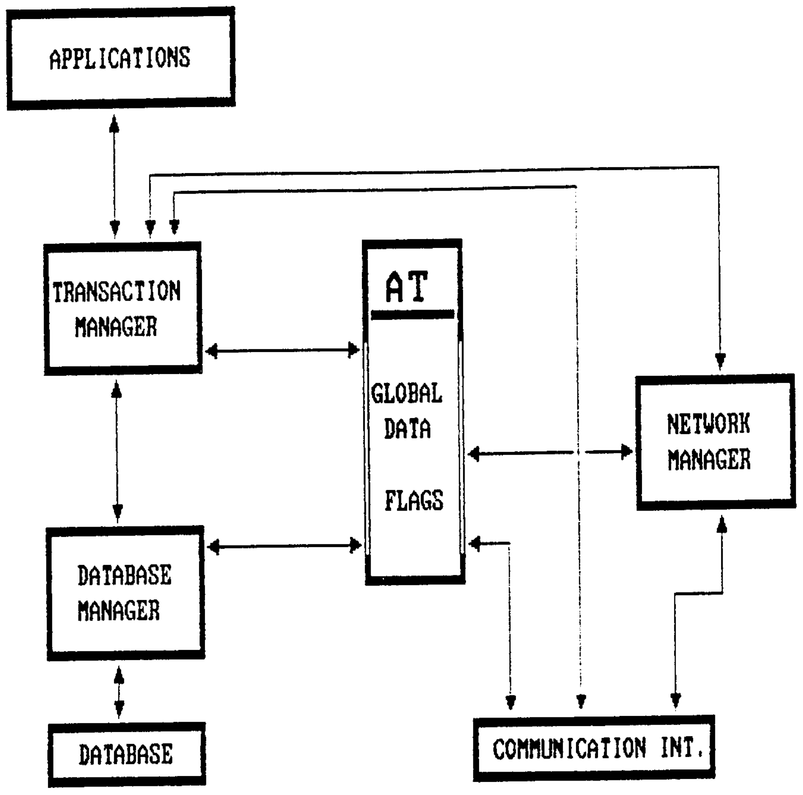

Figure 2. Node Functional Architecture

APPENDIX

ALLOCATION OF PRIMARY AND SECONDARY PARTITION NUMBERS

A typical distributed processing system consists of a number of computer nodes connected by a medium to form a network. The system could employ a partially replicated database (one where portions are replicated on different nodes) or a fully replicated distributed database at each node. For both cases, in order that an available system be maintained, even if in a somewhat degraded form, a mechanism is required to keep track of nodes and network partitions in the face of component failure, network damage, and network repair.

The complete network can be considered as the initial primary network before any actual partitioning occurs. A secondary partition which has divided from the initial primary becomes a new primary from which other secondary partitions can sub-divide. A primary partition can be considered the parent of the secondary partition, which is the child. The total number of levels of partitions will be limited in practice due to such considerations as the number of nodes in the network and the available data storage at each node for holding recovery information.

Each node must upkeep its own view of the network. The view is kept in an Availability Table (AT). The AT includes the following identification data for every node:

- Node Identifier (NI)
- Node Availability (NA)
- Primary Partition Number (PPN)
- Secondary Partition Number (SPN)

The node with the smallest NI in a network or network partition elects to become a coordinator for the segment. The coordinators for the secondary partition and its primary, control any recovery process.

The NA shows the node state, "available" or "not available" for transaction processing, in the primary partition.

The PPN identifies the partition to which the coordinator belongs while, the SPN identifies a node or node group which has been partitioned off from the primary segment as a result of a failure or damage situation.

PPN's must both uniquely identify a partition and also point to their parent primary partition. This means they must contain information of both the higher level parent partition as well as the secondary partition. PPNs therefore need to be sub-divided into a number of parts, each distinguishing a partition level. The most right-hand portion identifies the local partition. A PPN is determined by the first node to become active in the case of the initial primary partition, or by the first node of a secondary partition to recognise that a partition has occurred.

SPNs are unique integer values identifying the secondary partition to which a node belongs. An SPN is allocated by a node to those node NIs which fail to return an acknowledgment during a reliable transaction process. When a node failure or partitioned group of nodes is recognised by a parent node, ie there has been no transaction acknowledgment from a node or a group of nodes which does not include a coordinator, the next available integer value is given to the SPNs associated with the relevant node NIs in the local AT. The node then must broadcast a copy of the new AT to all other available nodes in the primary partition (REORG). Each available node which receives the REORG must initiate a benchmark and log for the new SPN.

Similarly, the first node at the new partition which recognises it has been partitioned off (no transaction acknowledgment from the coordinator node of its currently held network view (local AT)), must calculate a new PPN for all the available nodes within the new partition. It does this by adding to the right hand end of the parent PPN the next available SPN integer value.

As an example, take an initial primary network consisting of six nodes, NI = 1 to 6 with all nodes having their PPN and SPN values set to 0. This network elects node 1 as the coordinator because it has the lowest NI. Now if a partition should occur making node 4 and node 5 unavailable, this will be detected by a primary node during a reliable transaction process. The node which detects the partitioning will check its local AT for any previous partitions, ie NIs with SPNs greater than "0". In this case there are no previous partitions, and the node will set the SPNs for NIs "4" and "5" to "1" and broadcast (REORG) a copy of the new AT to all other available primary nodes. Nodes 1,2,3 and 6 will initiate benchmarks and logs.

The two nodes partitioned off will become a new primary partition with a PPN of "0;1". Node 4 and node 5 will continue to communicate between themselves.

One of the three possible actions listed below can now take place in any order:

a. either node 4 or node 5 will broadcast a NODE message.

b. either node 4 or node 5 during a periodic background activity, will check the current AT to determine if a coordinator election is required.

c. Either node 4 or node 5 will initiate a reliable transaction.

If "a" occurs first, then no change will occur. The local AT at both node 4 and node 5 will show all nodes available before and after a NODE.

If "b" occurs first, since all nodes appear as available at this stage, neither node 4 or node 5 will elect to be a coordinator.

If "c" occurs first then an acknowledgment will only be received from one node, Node 4 or node 5, even though one would be expected from five nodes. Nodes 1, 2, 3, and 6 will be treated as failed and marked as unavailable and a new AT will be broadcast (REORG message) to all available nodes, only node 4 and node 5. Both nodes will initiate benchmarks and logs for the current PPN, the parent primary with a PPN of "0", and then calculate their new PPN (0;1). Now when "b" occurs node 4 will elect to become a coordinator, because the secondary partition (nodes 4 and 5), does not have a coordinator.

DISTRIBUTION

	No. of Copies
Defence Science and Technology Organisation	
Chief Defence Scientist)	
Central Office Executive)	1 shared copy
Counsellor, Defence Science, London	Cont Sht
Counsellor, Defence Science, Washington	Cont Sht
Senior Defence Scientific Adviser	1 copy
Scientific Adviser POLCOM	1 copy
Navy Office	
Navy Scientific Adviser	1 copy
Director, Naval Combat Systems Engineering	1 copy
Army Office	
Scientific Adviser, Army	1 copy
Airforce Office	
Air Force Scientific Adviser	1 copy
Defence Intelligence Organisation	
Assistant Secretary Scientific Analysis	1 copy
Mr P. Whitbread, SRS DIO Liaison, Intelligence Systems Group, ITD	1 copy
Department of Defence	
Director General, Communications and Information Systems	1 copy
Aeronautical & Maritime Research Laboratory	
Director	1 copy
Chief Air Operations Division	Cont Sht
Chief Maritime Operations Division	Cont Sht
Electronics & Surveillance Research Laboratory	
Director	1 copy
Chief Information Technology Division	1 copy
Chief Electronic Warfare Division	Cont Sht
Chief Guided Weapons Division	Cont Sht
Chief Communications Division	Cont Sht
Chief Land, Space and Optoelectronics Division	Cont Sht
Chief High Frequency Radar Division	Cont Sht
Chief Microwave Radar Division	Cont Sht
Research Leader Command & Control and Intelligence Systems	1 copy
Research Leader Military Computing Systems	1 copy

DISTRIBUTION

	No. of Copies
Research Leader Command, Control and Communications	1 copy
Manager Human Computer Interaction Laboratory	Cont Sht
Head, Program and Executive Support	Cont Sht
Head Software Engineering Group	Cont Sht
Head, Trusted Computer Systems Group	Cont Sht
Head, Command Support Systems Group	1 copy
Head, Intelligence Systems Group	Cont Sht
Head, Systems Simulation and Assessment Group	Cont Sht
Head, Exercise Analysis Group	Cont Sht
Head, C3I Systems Engineering Group	1 copy
Mr A. Allwright, C3I Systems Engineering Group	1 copy
Head, Computer Systems Architecture Group	Cont Sht
Head, Information Management Group	Cont Sht
Mr D. O'Dea, Information Management Group	1 copy
Head, Information Acquisition & Processing Group	Cont Sht
Mr J. Schapel, Information Acquisition & Processing Group, ITD	1 copy
Author	1 copy
Publications & Publicity Officer ITD	1 copy
Libraries and Information Services	
Australian Government Publishing Service	1 copy
Defence Central Library, Technical Reports Centre	1 copy
Manager, Document Exchange Centre, (for retention)	1 copy
National Technical Information Service, United States	2 copies
Defence Research Information Centre, United Kingdom	2 copies
Director Scientific Information Services, Canada	1 copy
Ministry of Defence, New Zealand	1 copy
National Library of Australia	1 copy
Defence Science and Technology Organisation Salisbury, Research Library	2 copies
Library Defence Signals Directorate Canberra	1 copy
British Library Document Supply Centre	1 copy
Parliamentary Library of South Australia	1 copy
The State Library of South Australia	1 copy
Spares	
Defence Science and Technology Organisation Salisbury, Research Library	6 copies

Department of Defence

DOCUMENT CONTROL DATA SHEET

			1. Page Classification Unclassified
			2. Privacy Marking/Caveat (of document) N/A
3a. AR Number AR-008-918	3b. Laboratory Number DSTO-TR-0059	3c. Type of Report Technical Report	4. Task Number 87/226.3
5. Document Date July 1994	6. Cost Code 803885	7. Security Classification *U Document; U Title; U Abstract S (Secret) C (Confi) R (Rest) U (Unclass) * For UNCLASSIFIED docs with a secondary distribution LIMITATION, use (L) in document box.	8. No. of Pages 30 9. No. of Refs. 13
10. Title MAINTAINING FULLY REPLICATED DISTRIBUTED DATABASES IN THE PRESENCE OF NETWORK OR NODE FAILURE AND REPAIR			
11. Author(s) S.J. Miller		12. Downgrading/Delimiting Instructions N/A	
13a. Corporate Author and Address Electronics & Surveillance Research Laboratory PO Box 1500, Salisbury SA 5108		14. Officer/Position responsible for Security:..........N/A... Downgrading:......N/A... Approval forRelease:..DESRL..	
13b. Task Sponsor Navy			

15. Secondary Release Statement of this Document

APPROVED FOR PUBLIC RELEASE

16a. Deliberate Announcement

No Limitation

16b. Casual Announcement (for citation in other documents)

[✓ X] No Limitation ☐ Ref. by Author , Doc No. and date only.

17. DEFTEST Descriptors	18. DISCAT Subject Codes
Computer Architecture Naval Tactical Data System Combat Information Centres Computer Networks Distributed Processing Computer Communications Battlefield Information Systems	

19. Abstract

This report addresses a number of problems associated with the recovery of fully replicated distributed database systems in the eventuality of site failures and network partitions.